Úvod do distribuovaných algoritmov

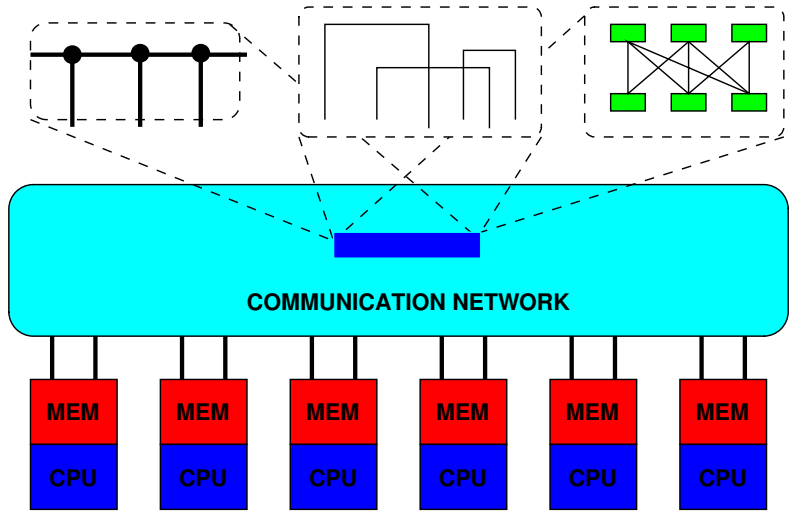
Rastislav Královič

Katedra informatiky,
FMFI UK,
Bratislava
kralovic@dcs.fmph.uniba.sk

zimný semester 2009

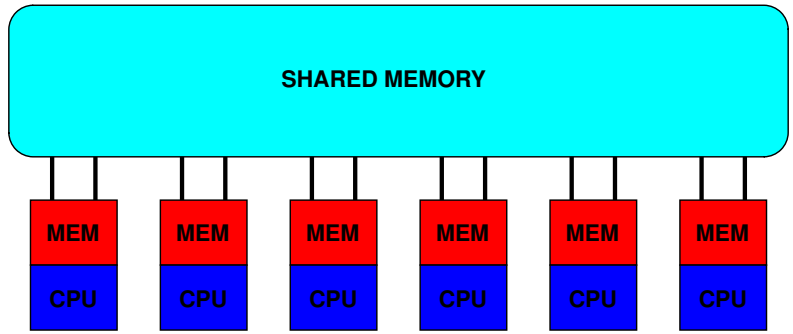
- **Gerard Tel:** *Introduction to Distributed Algorithms*, Cambridge University Press, 2000, ISBN 0521794838
- **Nancy Lynch:** *Distributed Algorithms*, Morgan Kaufmann Publishers, 1996, ISBN 1558603484
- **Frank Thomson Leighton:** *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*, Morgan Kaufmann Publishers, 1991, ISBN 1558601171

Distribuované výpočty



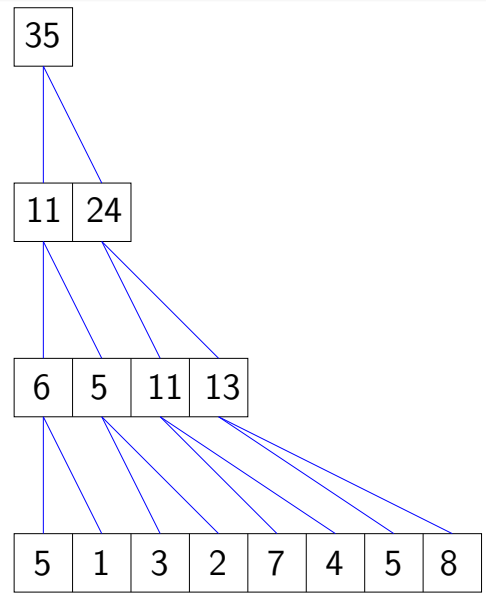
- kedy pomáha paralelný výpočet (efektivita, udržovateľnosť, ...)
- kde sú limity
- výpočet vs. komunikácia
- veľa parametrov $\Rightarrow$ ZOO modelov

PRAM



- abstrahuje od komunikácie (zdieľaná pamäť), zaujíma nás výpočet
- synchrónne procesory
- identifikátory

```
global read( A(i), a )
global write( a, B(i) )
for h = 1 to log n do
    if (i ≤ n/2^h)
        global read( B(2i − 1), x )
        global read( B(2i), y )
        z := x + y
        global write(z, B(i))
if i = 1 global write( z, S )
```

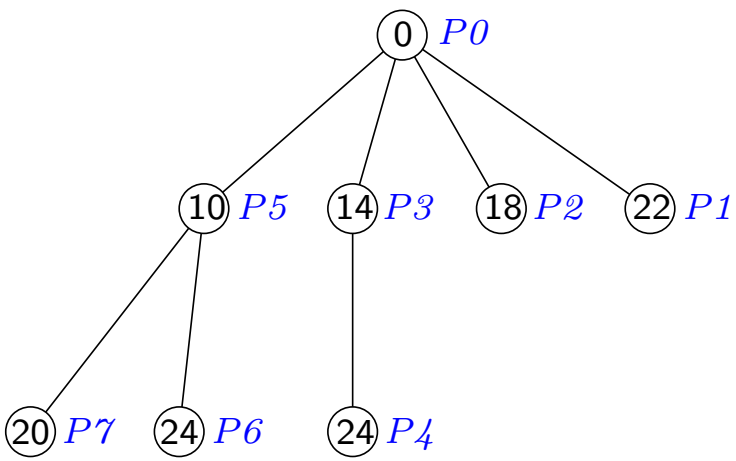


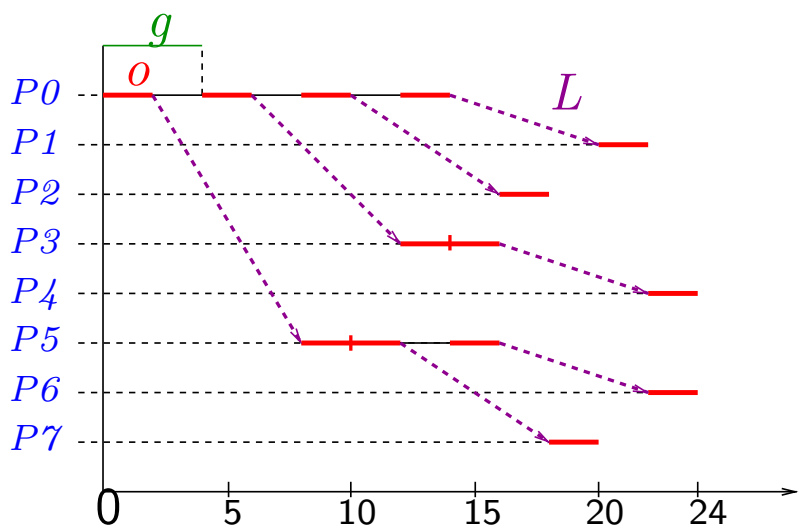
LogP model

- asynchrónny model
- komunikácia pomocou správ
- abstrahuje od štruktúry siete
- parametre:
  - L: horné ohraničenie latencie – zdržanie v sieti
  - o: overhead – zdržanie v procesore pri každej komunikácii
  - g: gap – minimálny čas medzi dvoma komunikáciami v procesore

poslať správu dĺžky m trvá $L \cdot m + 2o$

Príklad: broadcast pre $P = 8$, $L = 6$, $g = 4$, $o = 2$





BSP (Bulk Synchronous Parallel)

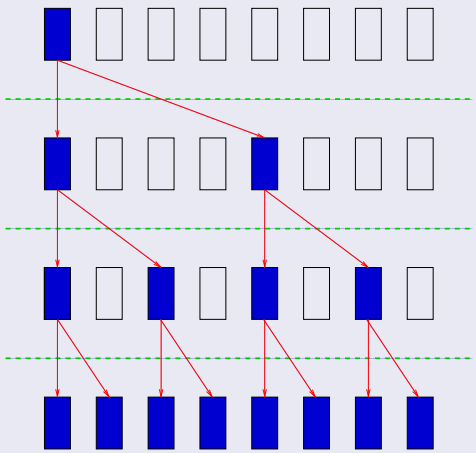
- sada procesorov spojených sieťou
- **superstep** pozostáva z
  - lokálny výpočet v každom procesore
  - komunikácia pomocou správ
  - barierová synchronizácia
- čas superstepu je $\max\{comp_i\} + \max\{comm_i\} \cdot g + l$
  - $comp_i$ je čas počítania i-teho procesora
  - $comm_i$ je objem dát prijatých a poslaných procesorom i
  - g je *lokálna priepustnosť* siete: výpočtový výkon / priepustnosť routera
  - l je čas potrebný na synchronizáciu

BSP broadcast – n hodnôt, p procesorov

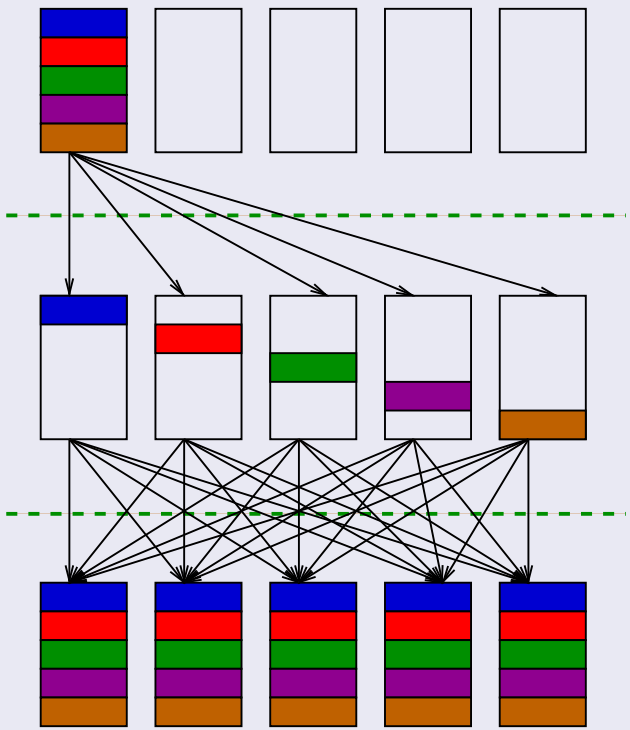
$$\max\{comp_i\} + \max\{comm_i\} \cdot g + l$$

jeden všetkým: cena $pn \cdot g + l$

strom:
cena $\leq \log p\,(3n \cdot g + l)$



dvojfázový: cena
$n \cdot g + l + \frac{n}{p}(p+1) \cdot g + l = O(n) \cdot g + O(1) \cdot l$



$\alpha\beta$ kalkulus

- použité v CM1, CM5 (CM LISP)
- distribuovaná dátová štruktúra: **xektor** – zoznam dvojíc {index$\mapsto$hodnota}
- α: konverzia hodnota$\mapsto$ konšt. xektor (t.j. zavedie hodnotu do všetkých procesorov)
  `(`α`+ '{a`$\mapsto$`1 b`$\mapsto$`2 c`$\mapsto$`3} '{a`$\mapsto$`3 b`$\mapsto$`3} ) `$\Rightarrow$` {a`$\mapsto$`4 b`$\mapsto$`5}`
- $\bullet$: ruší α
  `(CONS A X)` $\Rightarrow$ `([a b c ]. [x y z])`
  α`(CONS `$\bullet$`A `$\bullet$`X)` $\Rightarrow$ `[(a.x) (b.y) (c.z)]`
  α`(+ (* `$\bullet$`x 2) 1)` $\equiv$ `(`α`+ (`α`* x `α`2) `α`1)`
- β redukcia: paralelne v log. čase
  `(`β`+ '{A`$\mapsto$`1 B`$\mapsto$`2 C`$\mapsto$`3})` $\Rightarrow$ `6` (ignorujú sa indexy)
- veľkosť xektora x: `(SQRT (`β`+ (`α`* x x)))`
  výpočet $\sum x_i^3$: `(`β`+ `α`(POW `$\bullet$`x 3))`

CSP (Communicating Sequential Processes)

- sémantika
- pozorovanie procesu: postupnosť udalostí
- proces: množina pozorovaní
- vytváranie procesov:
  - **prefix**: $(x \mapsto P)$
  - **rekurzia**: `CLOCK = (tick↦CLOCK)`
    $F \equiv \mu X.F(X)$
  - **deterministický výber**: `(x↦P | y↦Q)`
  - **paralelizmus**: `(P || Q)` všetky "premiešania" pozorovaní
- **komunikácia**: zdieľaním udalostí

```
K = μ X.minca↦(káva↦X | čaj↦X)
Z = μ X.( čaj↦X | káva↦X | minca↦čaj↦X)
(Z || K) = μ X.(minca↦čaj↦X)
```

MapReduce (Google)

- masívny paralelizmus
- jednoduchý výpočtový model ⇒ automatická paralelizácia

- `map(in_key, in_value) -> list(out_key, intermediate_value)`
- `reduce(out_key, list(intermediate_value)) -> list(out_value)`

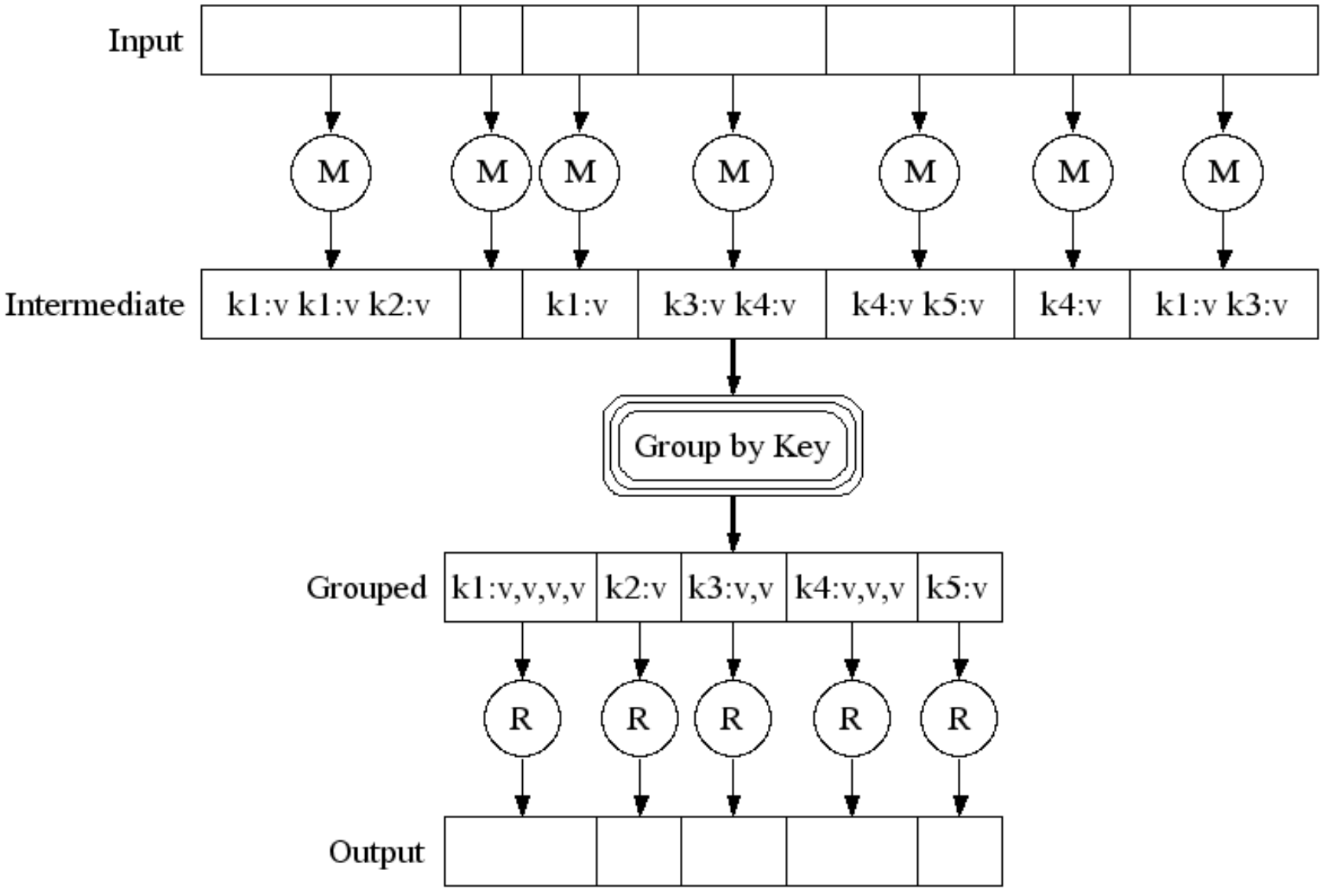
frekvencie slov

```
map(String input_key, String input_value):
    // input_key: document name
    // input_value: document contents
    for each word w in input_value:
      EmitIntermediate(w, "1");

reduce(String output_key, Iterator intermediate_values):
    // output_key: a word
    // output_values: a list of counts
    int result = 0;
    for each v in intermediate_values:
      result += ParseInt(v);
    Emit(AsString(result));
```

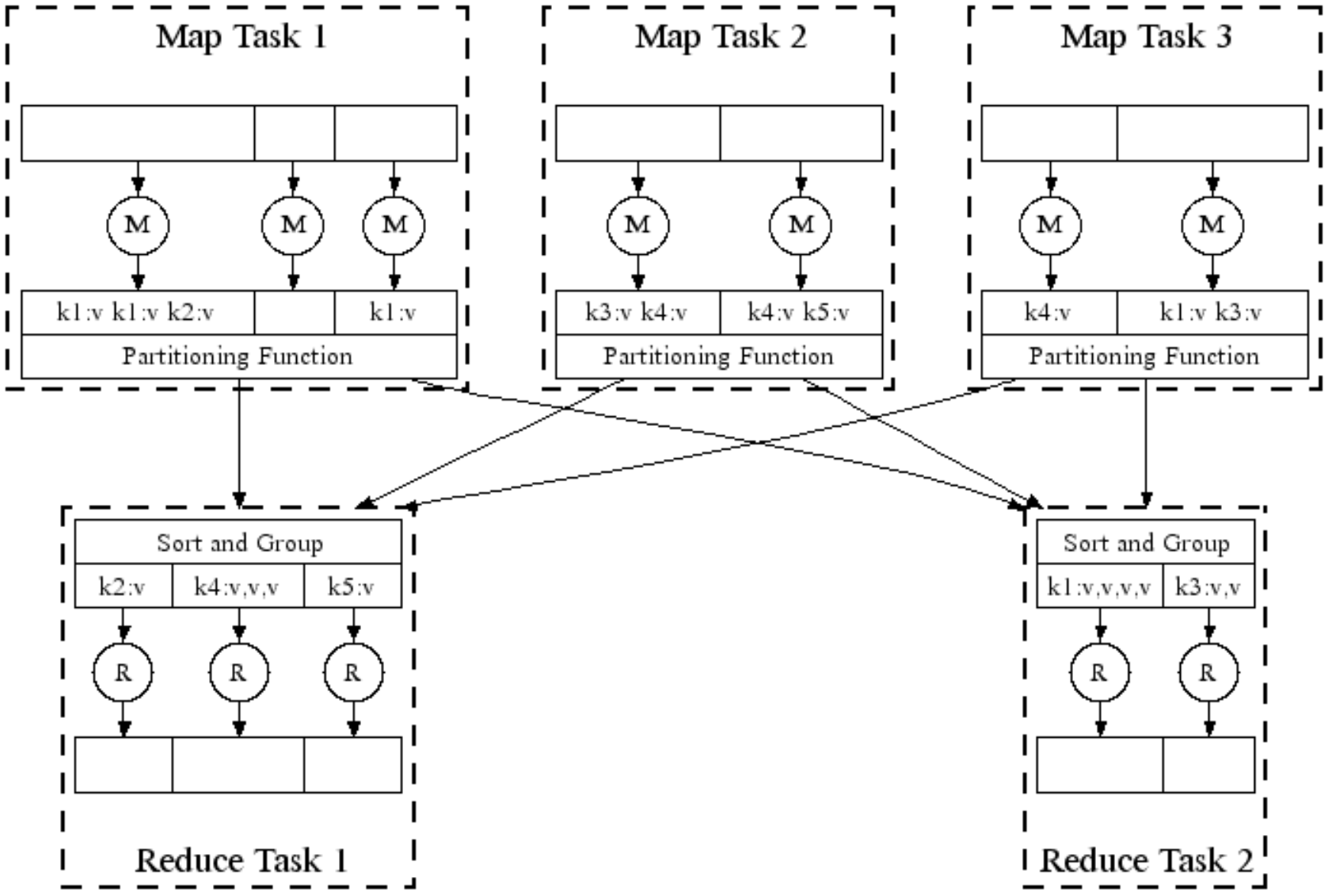
MapReduce (Google)

výpočet



MapReduce (Google)

paralelizácia



Sieťový model

- nezávislé zariadenia (procesy, procesory, uzly)
- posielanie správ
- lokálny pohľad (číslovanie portov)
- asynchrónne, spoľahlivé správy
- topológia siete
- wakeup/terminácia

zložitosť: v závislosti od počtu procesorov

- počet správ / bitov
- čas (beh normovaný na max. dĺžku 1)

horný odhad pre problém

Existuje algoritmus, ktorý **pre všetky** topológie, vstupy, časovania, ...
... pracuje správne a vymení najviac $f(n)$ správ

dolný odhad pre problém

Pre každý algoritmus, **existuje kombinácia** topológie, vstupu, časovania, ...
... že buď nepracuje správne alebo vymení aspoň $f(n)$ správ

Sieťový model: broadcast a convergecast (maximum)

const: *deg* : integer
ID : integer
Neigh : [1...*deg*] link
parent : link
var: *msg* : text

Init:

if *parent* $=$ *NULL*
  send $\langle$**dispatch**, *msg*$\rangle$ **to self**

Code:

loop forever

On receipt $\langle$**dispatch**, *new_msg*$\rangle$ from *parent* or self :

msg $:=$ *new_msg*
for all *l* $\in$ *Neigh* $-$ {*parent*} **do**
  send $\langle$**dispatch**, *msg*$\rangle$ **to** *l*
skonči algoritmus

const: *deg* : integer
ID : integer
Neigh : [1...*deg*] link
parent : link
msg : integer
var: *count* : integer
max : integer

Init:

count $=$ 0
max $=$ *msg*
if *deg* $=$ 1
  send $\langle$**my_max**, *max*$\rangle$ **to** *parent*

Code:

loop forever

On receipt $\langle$**my_max**, *x*$\rangle$ from *Neigh*[*i*]:

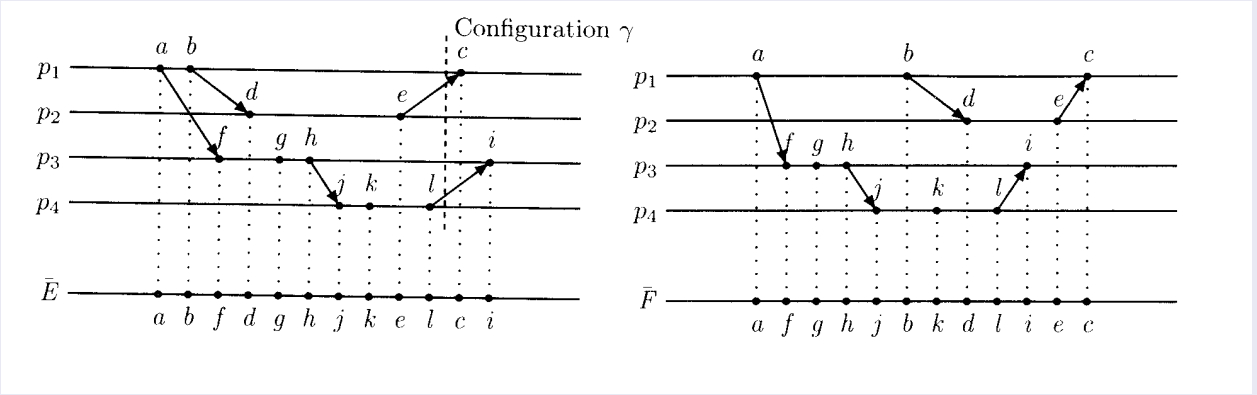
max $=$ maximum{*max*, *x*}
count $++$
if *count*$=$ *deg* $-$ 1
  send $\langle$my_max, *max*$\rangle$ **to** *parent*
  skonči algoritmus

Sieťový model

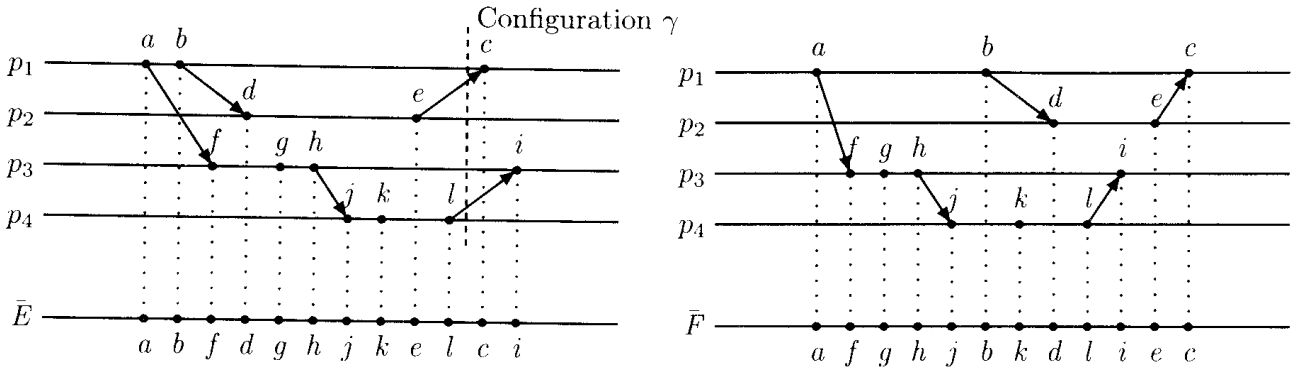
- nezávislé zariadenia (procesy, procesory, uzly)
- posielanie správ
- lokálny pohľad (číslovanie portov)
- asynchrónne, spoľahlivé správy
- topológia siete
- wakeup/terminácia

formálny model: prechodové systémy

- systém je trojica $(\mathcal{C}, \mapsto, \mathcal{I})$
- beh (execution) je postupnosť $\gamma_0 \mapsto \gamma_1 \mapsto \gamma_2 \mapsto \cdots$, kde $\gamma_0 \in \mathcal{I}$
- $\mathcal{C} = \mathcal{C}_L^n \times \mathcal{M}$: stavy systému = stavy procesorov + správy na linkách
- $\mapsto$: krok jedného procesora (výpočet, poslanie správy, prijatie správy)
- fair scheduler?



Sieťový model



formálny model: prechodové systémy

- (ne)závislosť udalostí:
  1) poslanie pred prijatím
  2) časovanie v rámci procesora
  3) tranzitivita
- výpočet (computation): trieda ekvivalencie behov

logické hodiny (Lamport)

var θ_p : integer **init** 0 ;

(* An internal event *)
$\theta_p := \theta_p + 1$;
Change state

(* A send event *)
$\theta_p := \theta_p + 1$;
send $\langle$ **messg**, $\theta_p \rangle$; Change state

(* A receive event *)
receive $\langle$ **messg**, $\theta \rangle$; $\theta_p := \max(\theta_p, \theta) + 1$;
Change state

cvičenia

- voľba šéfa na (anonymných) stromoch
- voľba šéfa na (anonymnej) mriežke